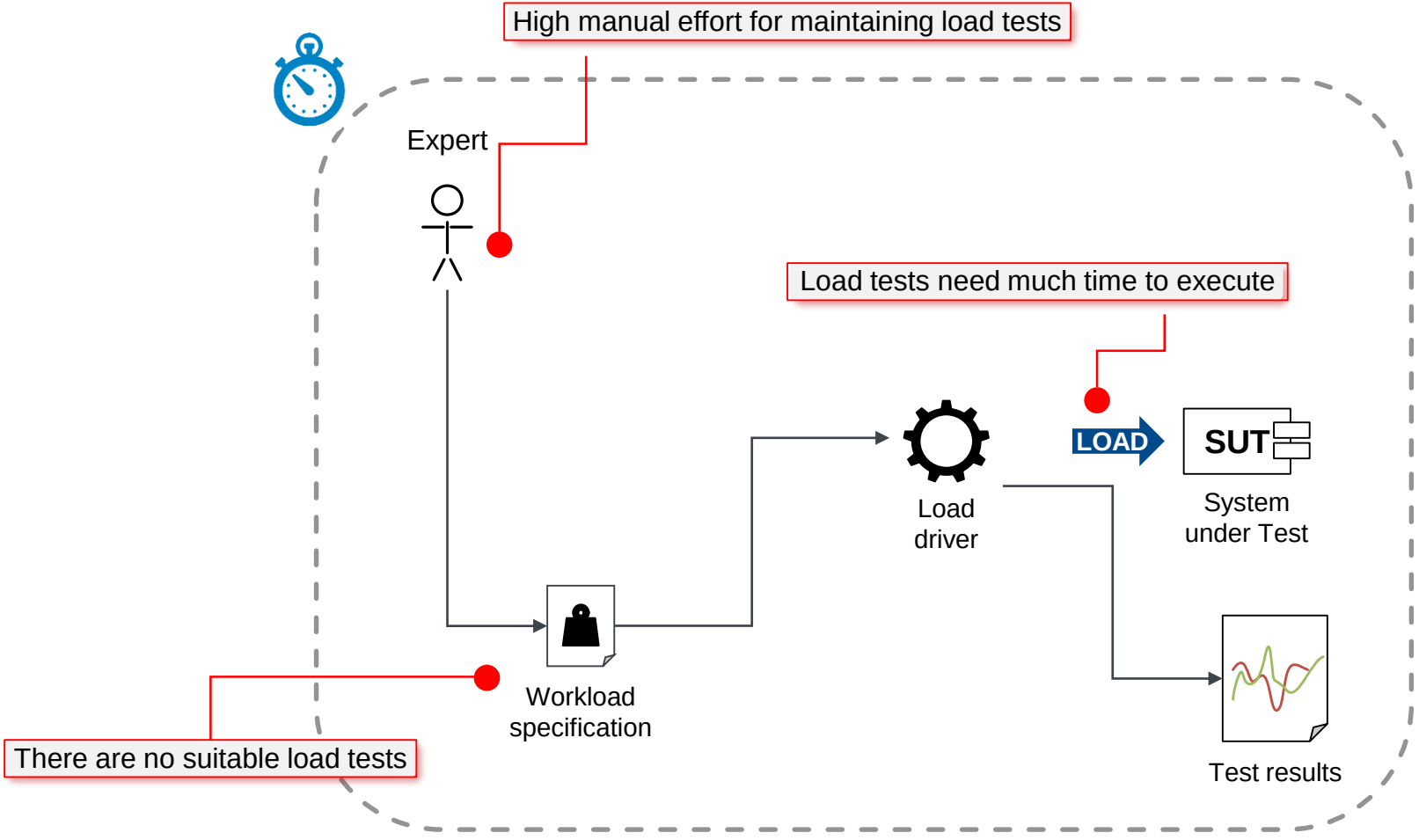
Behavior-driven Load Testing Using Contextual Knowledge
– Approach and Experiences –

Henning Schulz, Dušan Okanović,
André van Hoorn @andrevanhoorn
Vincenzo Ferme, Cesare Pautasso

10th ACM/SPEC Int. Conf. on Performance Engineering (ICPE 2019)
Mumbai, India – April 11, 2019

University of Stuttgart NOVATEC KIRATECH WE DEVOPS IT
SPONSORED BY THE Federal Ministry of Education and Research DFG

Load Testing – Approach and Challenges

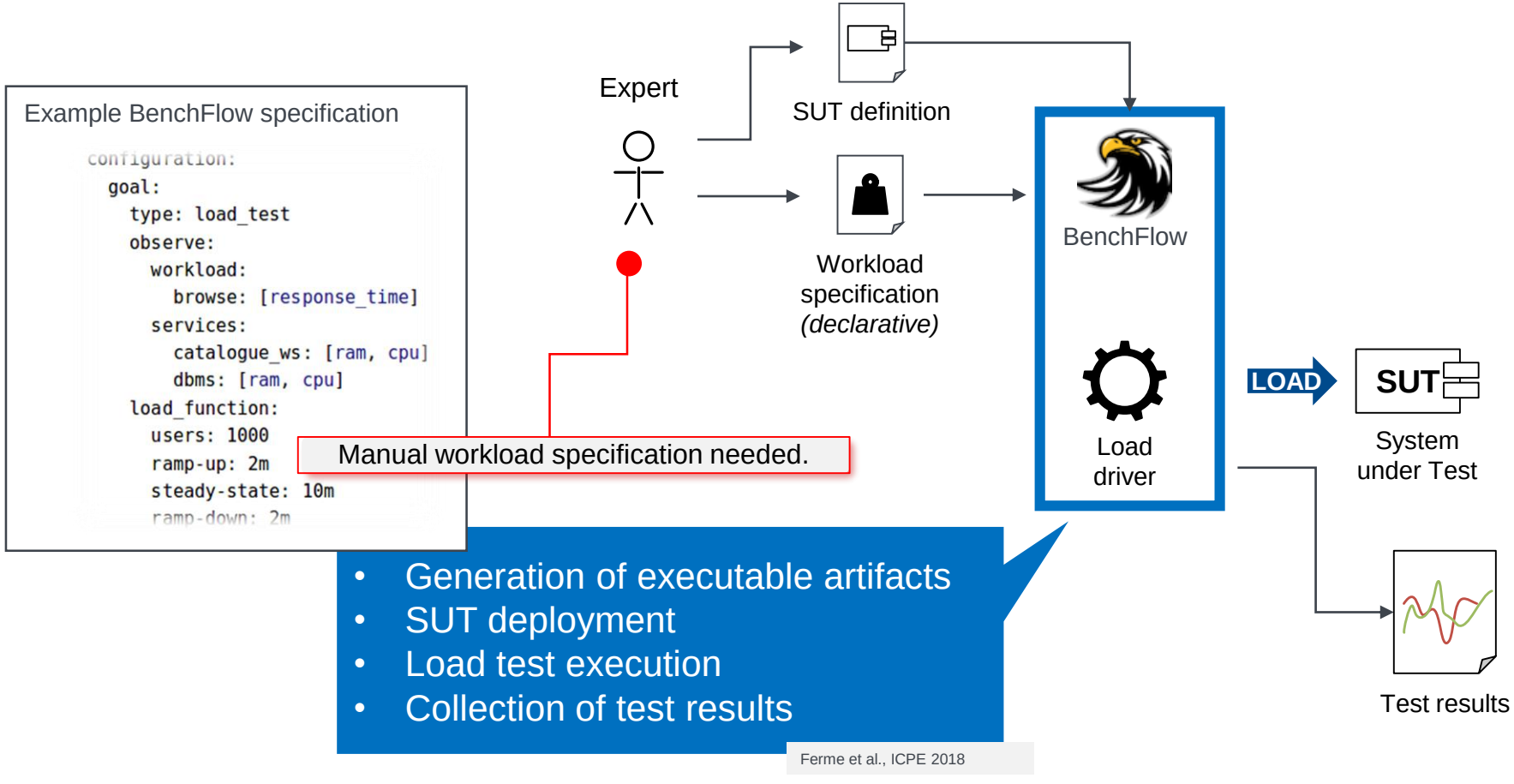


BenchFlow – Declarative Load Testing

Background

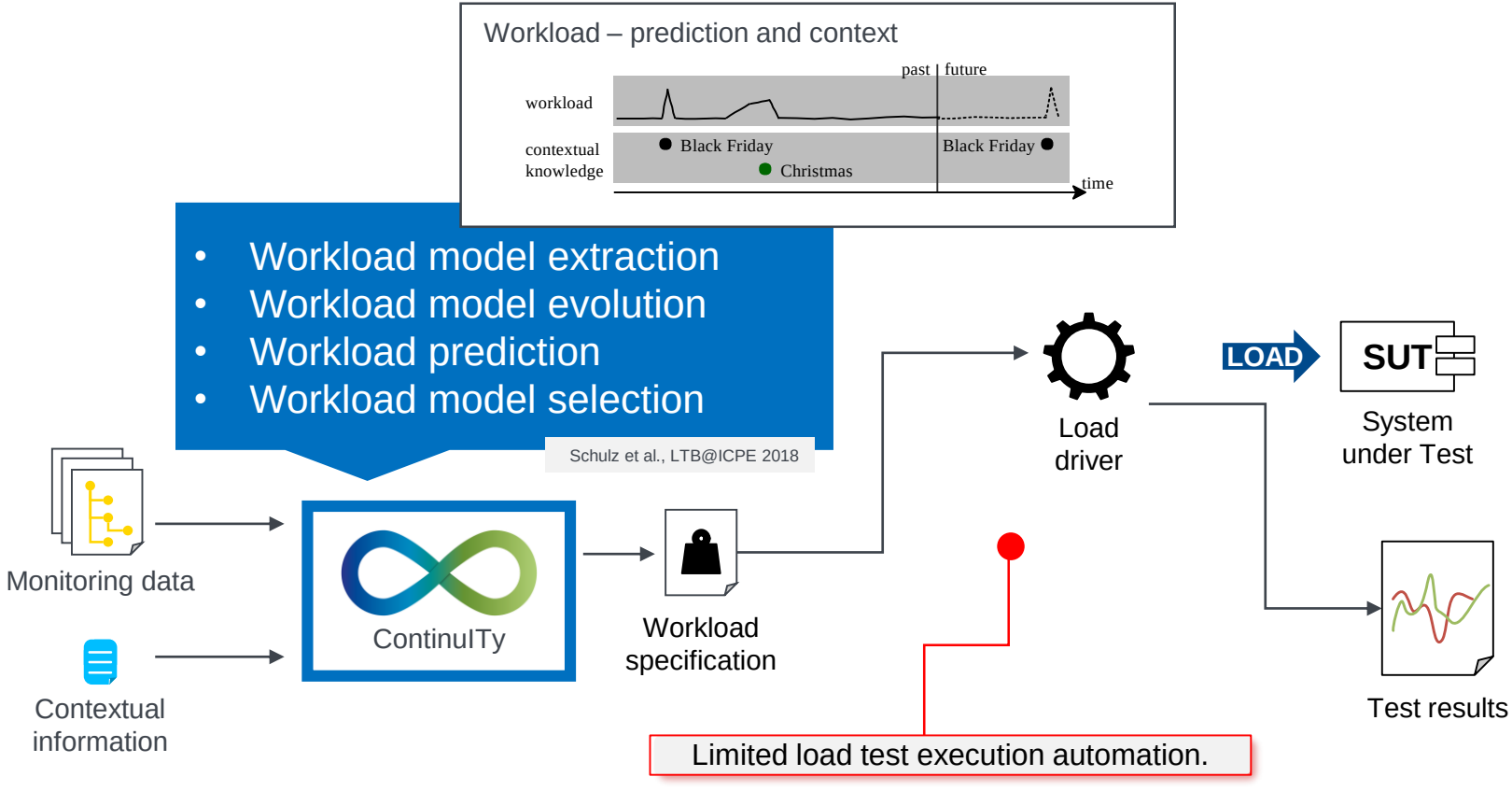
Example BenchFlow specification

```
configuration:
  goal:
    type: load_test
    observe:
      workload:
        browse: [response_time]
      services:
        catalogue_ws: [ram, cpu]
        dbms: [ram, cpu]
  load_function:
    users: 1000
    ramp-up: 2m
    steady-state: 10m
    ramp-down: 2m
```



ContinuITy – Continuous Load Testing in DevOps

Background



Behavior-driven (Functional) Testing – Workflow

Background

1. Write story

Plain text

```
Scenario: A trader is alerted of status

Given a stock and a threshold of 15.0
When stock is traded at 5.0
Then the alert status should be OFF
```

2. Map steps to Java

POJO

```
public class TraderSteps {
    private TradingService service;   // Injected
    private Stock stock; // Created

    @Given("a stock and a threshold of $threshold")
    public void aStock(double threshold) {
        stock = service.newStock("STK", threshold);
    }
    @When("the stock is traded at price $price")
    public void theStockIsTraded(double price) {
        stock.tradeAt(price);
    }
    @Then("the alert status is $status")
    public void theAlertStatusIs(String status) {
        assertThat(stock.getStatus().name(), equalTo(status));
    }
}
```

3. Configure Stories

Only once

```
public class TraderStories extends JUnitStories {

    public Configuration configuration() {
        return new MostUsefulConfiguration()
            .useStoryLoader(new LoadFromClasspath(this.getClass()))
            .useStoryReporterBuilder(new StoryReporterBuilder()
                .withCodeLocation(codeLocationFromClass(this.getClass()))
                .withFormats(CONSOLE, TXT, HTML, XML));
    }

    public List<CandidateSteps> candidateSteps() {
        return new InstanceStepsFactory(configuration(),
            new TraderSteps(new TradingService())).createCandidateSteps();
    }

    protected List<String> storyPaths() {
        return new StoryFinder().findPaths(codeLocationFromClass(this.getClass()),
"**/*.story");
    }
}
```

4. Run Stories

With any of

5. View Reports

HTML

Scenario: A trader is alerted of status

Given a stock and a threshold of 15.0
When stock is traded at 5.0
Then the alert status is OFF

Changes

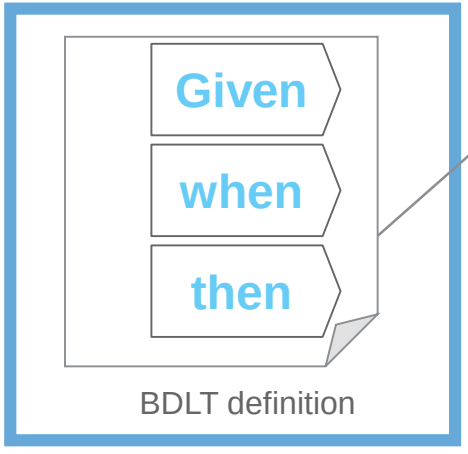
Given **When** **Then**

Initial context — Outcome

Examples © jbehave

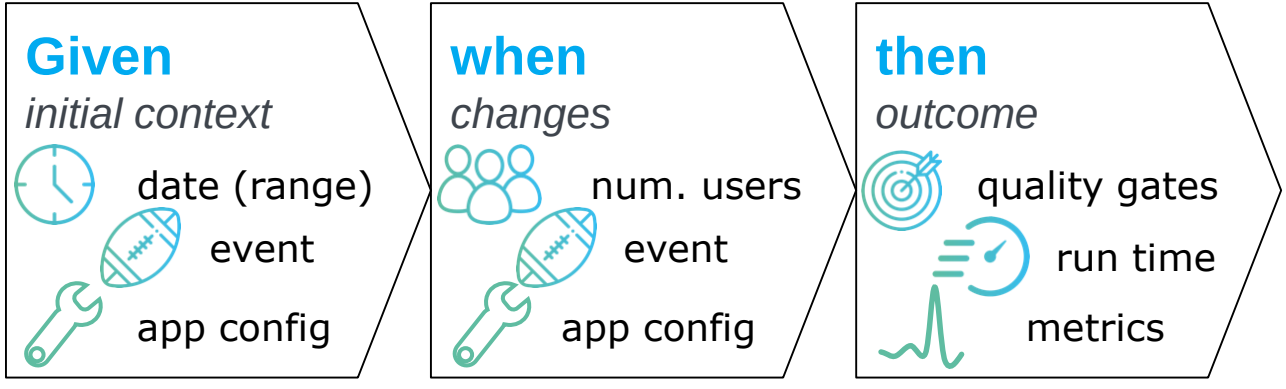
Behavior-driven *Load* Testing (BDLT) – Example

Approach



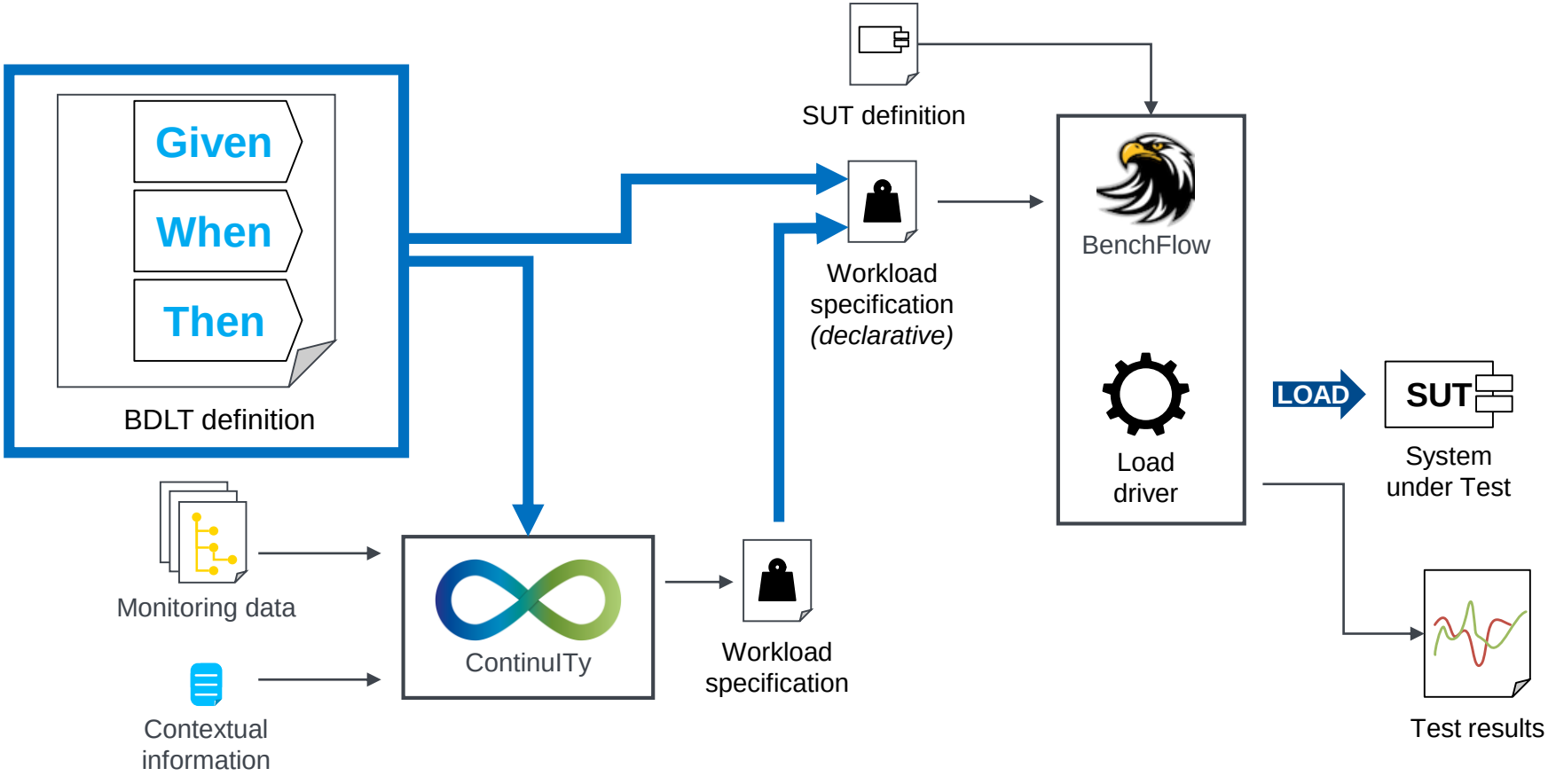
Given the next Black Friday,
when varying the CPU cores between 1 and 4,
then run the experiment for 1h
and
ensure the maximum CPU utilization is less than 60%.

BDLT example definition

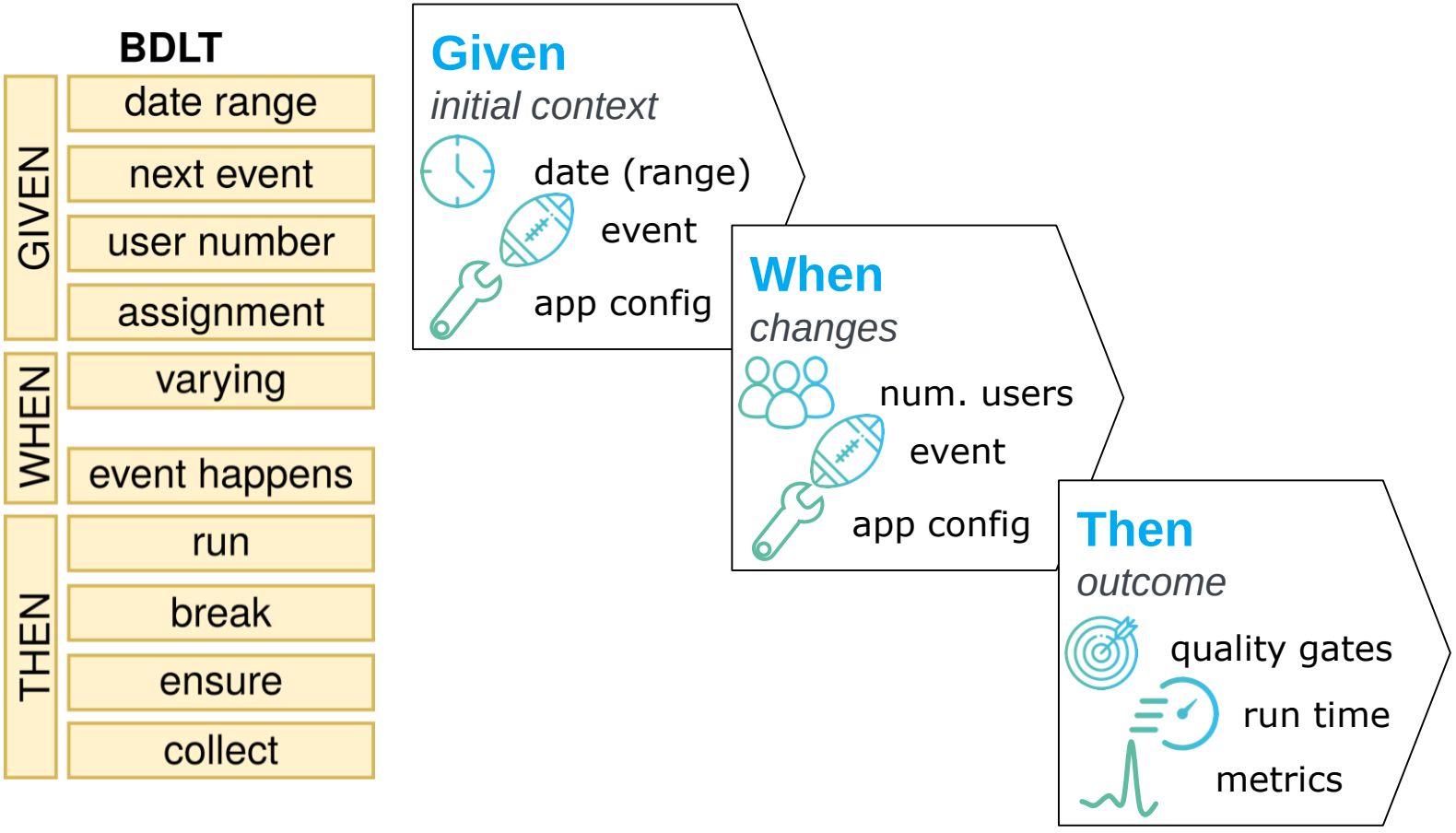


Behavior-driven *Load* Testing (BDLT) – Workflow

Approach

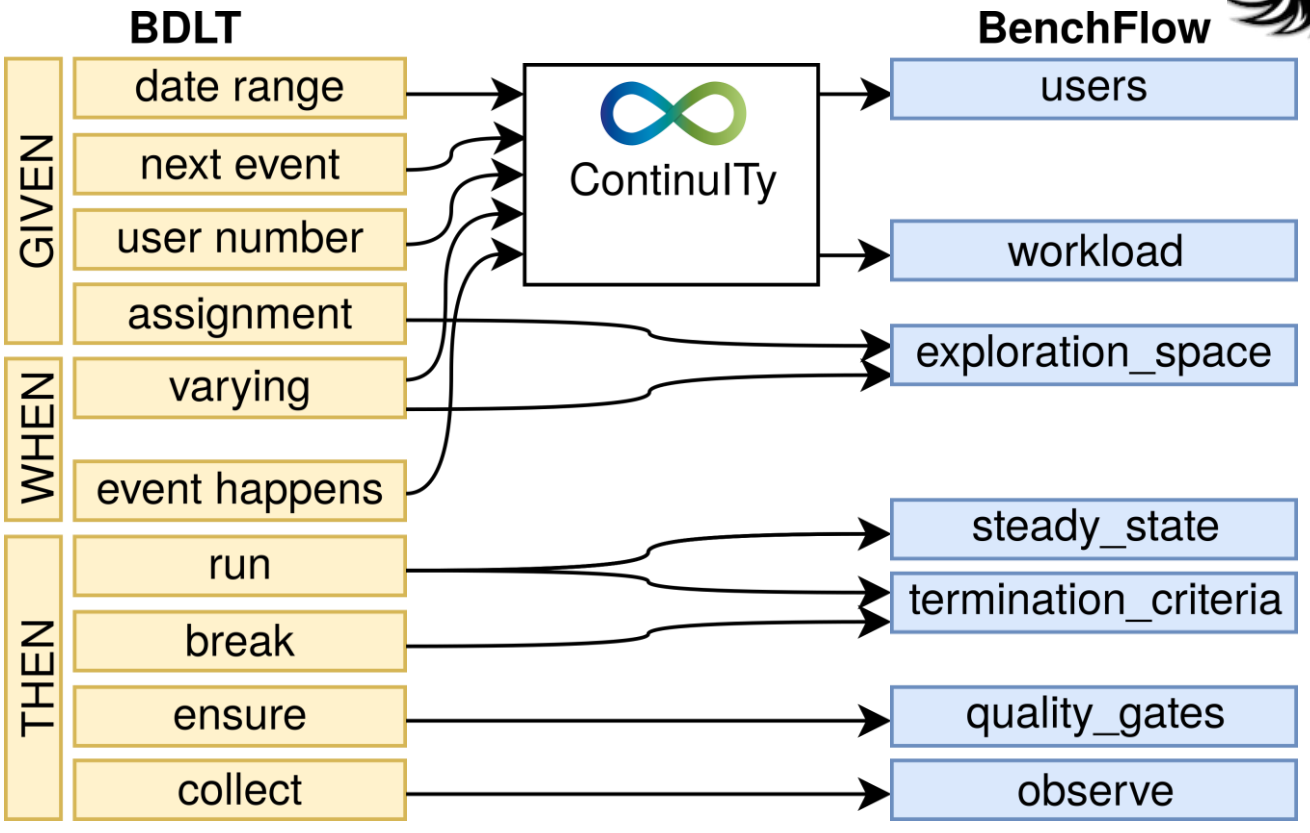


BDLT Language Definition



EBNF-based BDLT grammar with extension points for events

Mapping from BDLT to ContinuITy and BenchFlow



Evaluation – Research Questions

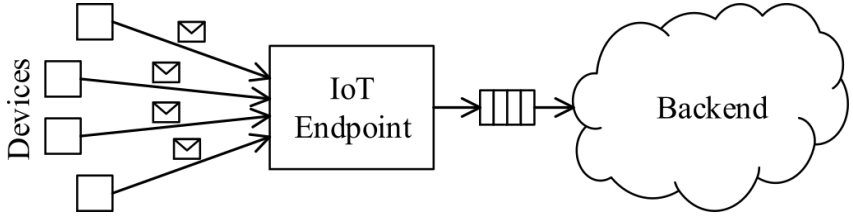
RQ1 How expressive is the BDLT language in regards to load test concerns of industrial use cases?

RQ2 How would BDLT be used in industrial contexts?

RQ3 What are the benefits and limitations of using BDLT in comparison to defining load test scripts?

Case Study – System and Method

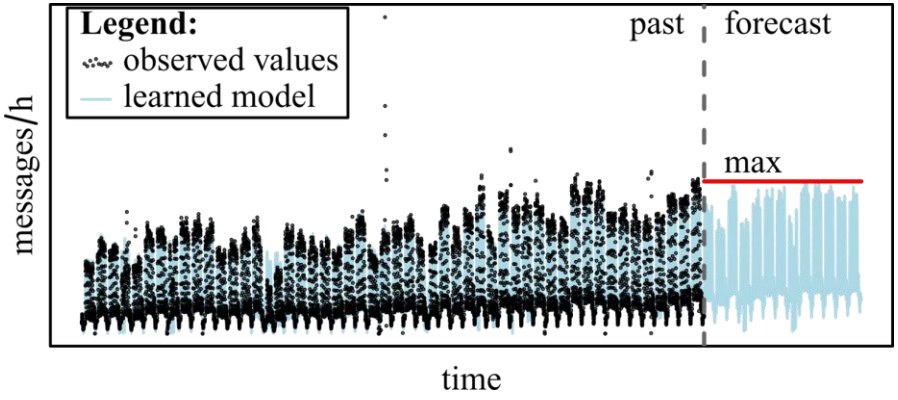
- System under study
  - IoT system from automotive sector
  - IoT endpoint migrating to Cloud
- Methodology
  - Workshops
  - We iteratively developed test specifications based on expert feedback
  - Incorporation of production monitoring data (per-device behavior and intensity)
- Four scenarios



Name	goal	quality_gates
configuration exploration	exhaustive_exploration	CPU load, message latency
continuous quality assurance	load	number of instances, cost
recovery spike	load	queue length
more devices	load	CPU load

Case Study – *Configuration Exploration*

Given the next three months
and
the number of users set to the maximum
when varying the CPU cores
between 0.5 and 4 in steps of 0.5
and
varying the number of instances
between 1 and 5
and
varying the RAM among (1GB, 2GB, 4GB)
then run each experiment for 1 hour
and
ensure the average CPU load is less than 15%
and
ensure the message latency is less than 2 seconds

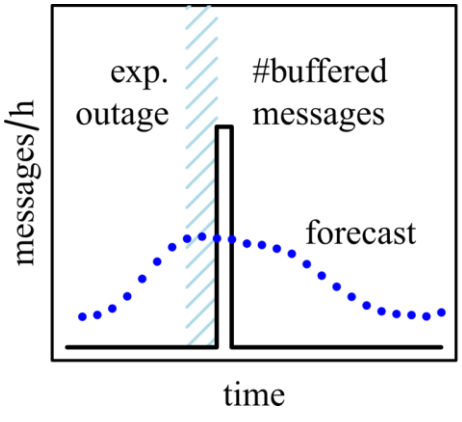


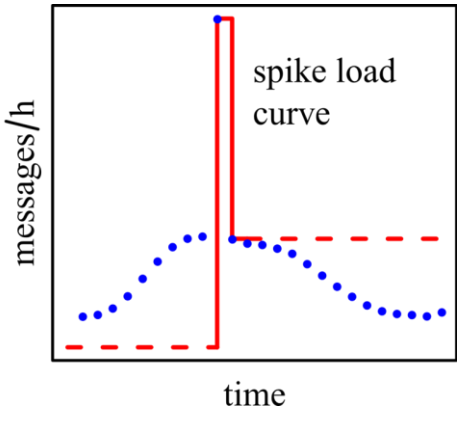
Case Study – *Recovery Spike*

Given 2018/10/15 9:00

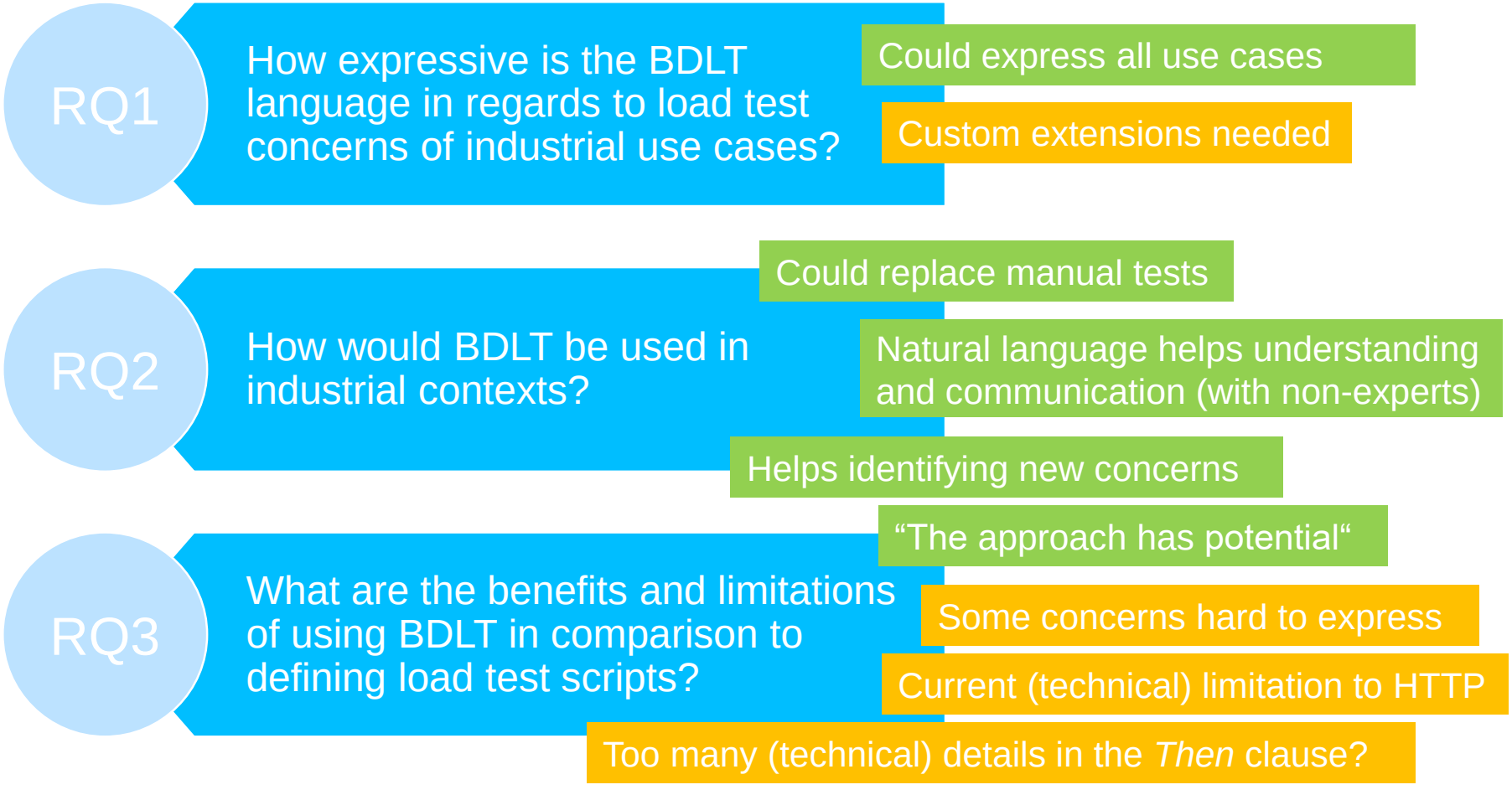
when an outage happened from
2018/10/15 7:00 to 2018/10/15 9:00

then run the experiment for 2 hours
and
ensure the final queue length is
less than 100

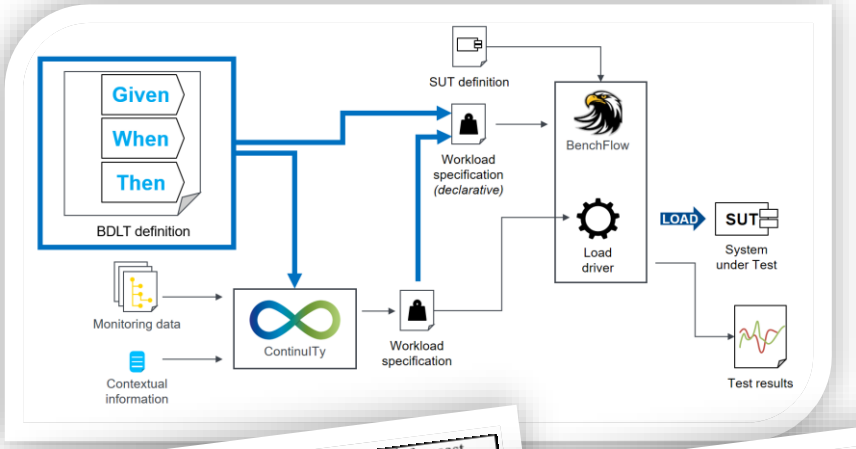


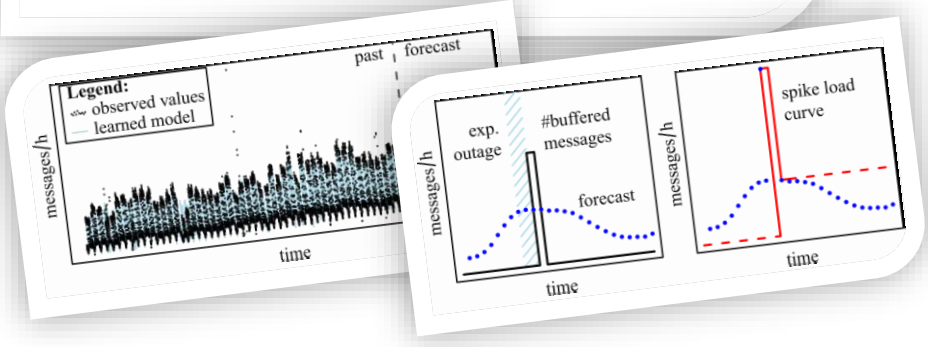


Evaluation – Lessons Learned



Summary





Future Work

- Extending the language
- Improving tool support
- Further evaluation
- Supporting regression queries
- Integration into agile development methods, e.g., user stories from Scrum tickets
- Integration into Declarative Performance Engineering landscape Walter et al., ICPE 2016

+ Additional (laboratory) case study Avritzer et al.,ECSA 2018

+ Supplementary material

Behavior-driven Load Testing Using Contextual Knowledge
– Approach and Experiences –

Henning Schulz, Dušan Okanović,
André van Hoorn @andrevanhoorn
Vincenzo Ferme, Cesare Pautasso

10th ACM/SPEC Int. Conf. on Performance Engineering (ICPE 2019)
Mumbai, India – April 10, 2019

University of Stuttgart NOVATEC USI KIRATECH WE DEVOPS IT
SPONSORED BY THE Federal Ministry of Education and Research DFG

References

Avritzer et al.,*ECSA* 2018

Alberto Avritzer, Vincenzo Ferme, Andrea Janes, Barbara Russo, Henning Schulz, André van Hoorn: *A Quantitative Approach for the Assessment of Microservice Architecture Deployment Alternatives by Automated Performance Testing.* ECSA 2018: 159-174

Ferme et al., ICPE 2018

Vincenzo Ferme, Cesare Pautasso: *A Declarative Approach for Performance Tests Execution in Continuous Software Development Environments.* ICPE 2018: 261-272

Schulz et al., LTB@ICPE 2018

Henning Schulz, Tobias Angerstein, André van Hoorn: *Towards Automating Representative Load Testing in Continuous Software Engineering.* ICPE Companion 2018: 123-126

Vögele et al., SoSyM 2016

Christian Vögele, André van Hoorn, Eike Schulz, Wilhelm Hasselbring, Helmut Krcmar: WESSBAS: extraction of probabilistic workload specifications for load testing and performance prediction - a model-driven approach for session-based application systems. Software and System Modeling 17(2): 443-477 (2018)

Walter et al., ICPE 2016

Jürgen Walter, André van Hoorn, Heiko Koziolek, Dusan Okanovic, Samuel Kounev: *Asking "What"?, Automating the "How"?: The Vision of Declarative Performance Engineering.* ICPE 2016: 91-94